REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
21-09-2010	Final Report	15-Apr-2007 - 14-Apr-2010

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Progressive Email Classifier (PEC) for ingress enterprise network traffic analysis	W911NF-07-1-0178
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER LDXXX2
6. AUTHORS	5d. PROJECT NUMBER
Jyh-Charn Liu	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES	8. PERFORMING ORGANIZATION REPORT NUMBER
Texas Engineering Experiment Station Research Services Texas Engineering Experiment Station College Station, TX 77845 -4645	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 52758-CS.2

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

This report summarizes research findings of the project “Progressive Email Classifier (PEC) for Ingress Enterprise Network Traffic Analysis “. We have developed a series of solutions which are designed to serve the needs of gateway level detection of spam like traffic, with and without prior defined patterns. The first major solution is the scoreboard architecture, which can track the scores and ages of patterns with a constant running time. Next, we developed a packetized processing software architecture, PFlex, for the regular expression pattern matcher Flex to

15. SUBJECT TERMS

traffic analysis, architecture, spam filter, statistics, pattern detection, regular expression, deep packet inspection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jyh-Charn Liu
UU	UU	UU	UU		19b. TELEPHONE NUMBER 979-845-8739

Report Title

Progressive Email Classifier (PEC) for ingress enterprise network traffic analysis

ABSTRACT

This report summarizes research findings of the project “Progressive Email Classifier (PEC) for Ingress Enterprise Network Traffic Analysis “. We have developed a series of solutions which are designed to serve the needs of gateway level detection of spam like traffic, with and without prior defined patterns. The first major solution is the scoreboard architecture, which can track the scores and ages of patterns with a constant running time. Next, we developed a packetized processing software architecture, PFlex, for the regular expression pattern matcher Flex to support packet level content scanning. The third major solution is a SA2PX tool which can translate SpamAssassin into Posix format, so that it can be ported to different platforms. The fourth major solution is a new Nondeterministic Finite Automata (NFA) algorithm for regular expression scanning, which can support overlapped matching, and can resolve matching ambiguity. We have tested these solutions in simulations, and run them on different computing platforms, including the multicore PC, the Bivio model 7500 DPI multicomputer, and FPGA. The solutions can be integrated into a system to supplement existing server-based spam filters by providing real-time statistics based spam information. The overall system design can be broadly expanded to support other network security functions.

List of papers submitted or published that acknowledge ARO support during this reporting period. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Number of Papers published in peer-reviewed journals: 0.00

(b) Papers published in non-peer-reviewed journals or in conference proceedings (N/A for none)

Number of Papers published in non peer-reviewed journals: 0.00

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts): 0

Peer-Reviewed Conference Proceeding publications (other than abstracts):

1. Sheng-Ya Lin, Jonas Tan, Jyh-Charn Liu, Michael Oehler, “High-Speed Detection of Unsolicited Bulk Emails”, the Symposium on Architectures for Networking and Communications Systems, Dec, 2007
2. Shi Pu, Cheng-Chung Tan and Jyh-Charn Liu, “SA2PX: A Tool to Translate SpamAssassin Regular Expression Rules to POSIX”, 6th Conference on Email and Anti-Spam, 2009
3. Hao Wang, Shi Pu, Gabe Kneze, Jyh-Charn Liu, “A Modular NFA Architecture for Regular Expression Matching”, accepted to ACM SIGDA FPGA conference, 2010

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts): 3

(d) Manuscripts

1. Cheng-Chung Tan, Jyh-Charn Liu, "Similarity-Based Sorting and Obfuscation Analysis of Message Streams", to be submitted
2. Shengya Lin, Jyh-Charn Liu, "On Classification of TCP Flows in the Middle of End-to-End Path", to be submitted
3. Hao Wang, Jyh-Charn Liu, "an NFA architecture for approximate string matching", under preparation
4. Cheng-Chung Tan, Jyh-Charn Liu, "Packetized string processing: architecture and performance", to be submitted

Number of Manuscripts: 4.00

Patents Submitted

Patents Awarded

Graduate Students

NAME	PERCENT_SUPPORTED
Shengya Lin	0.50
C.C. Tan	0.50
Y-J Chang	0.25
Pu Duan	0.50
Hong Lu	0.20
Gabriel Knezek	0.25
Hao Wang	0.25
Shi Pu	0.25
FTE Equivalent:	**2.70**
Total Number:	**8**

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Udo Pooch	0.05	No
Jyh-Charn Liu	0.20	No
FTE Equivalent:	**0.25**	
Total Number:	**2**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
Bradley William Reitmeyer	0.10
FTE Equivalent:	**0.10**
Total Number:	**1**

Student Metrics
This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	1.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	1.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:	0.00

Names of Personnel receiving masters degrees

NAME

Total Number:

Names of personnel receiving PHDs

NAME
Hong Lu

Total Number: 1

Names of other research staff

NAME	PERCENT_SUPPORTED

FTE Equivalent:
Total Number:

Sub Contractors (DD882)

Inventions (DD882)

Summary

This report summarizes research findings of the project "Progressive Email Classifier (PEC) for Ingress Enterprise Network Traffic Analysis ". We have developed a series of solutions which are designed to serve the needs of gateway level detection of spam like traffic, with and without prior defined patterns. The first major solution is the scoreboard architecture, which can track the scores and ages of patterns with a constant running time. Next, we developed a packetized processing software for Flex to support packet level content scanning, called PFlex. The third major solution is a SA2PX tool which can translate SpamAssassin into Posix format, so that it can be ported to different platforms. The fourth major solution is a new Nondeterministic Finite Automata (NFA) algorithm for regular expression scanning, which can support overlapped matching, and can resolve matching ambiguity. We have tested these solutions in simulations, and run them on different computing platforms, including the multicore PC, the Bivio model 7500 DPI multicomputer, and FPGA. The solutions can be integrated into a system to supplement existing server-based spam filters by providing real-time statistics based spam information. The overall system design can be broadly expanded to support other network security functions.

Scoreboard Architecture [1]

The main objective of PEC is to develop more effective ways to identify flooding of spam at the gateway, so that they can be intercepted or quarantined before reaching end users. Despite the rich collection of signatures and rules in existing spam filters, they are often one step behind the fast flux tactics of spammers. To be positioned at the ingress of an enterprise network, PEC needs to detect freshly crafted spam, or *unsolicited bulk email (UNBE)*, that have not been seen by any deployed anti-spam tools. PEC is designed to detect anomalous surges of *feature instances* (FI) of major spam at minimal computing costs. Computing cost is a critical design factor in order to handle the large volume of traffic, and scalability of the solution.

An FI is a particular realization of the *UNBE feature* F, which is any email construct that is likely to be used by spammers. Formally speaking, F ={ α_1, α_2, α_3 ..., } represents a set of binary strings which can be expressed and parsed by a finite automata, and each of $\alpha_i \in$ F is an FI of F. An email construct is not a viable spam feature if it cannot be effectively used to discriminate regular emails from spam, e.g., the greeting words, subject line, etc.

Let γ denote a newly identified FI by the feature parser, γ is assigned one of three *states*: $X_\gamma \leftarrow$ G/B/W, i.e., *Gray* (unchecked), *Black* (UNBE), or *White* (not UNBE), until it is removed from the system. X_γ (*v*) $\leftarrow$ G, where *v* is the current VC value. γ will be retained for a certain time period before its state changes, i.e., $X_\gamma \leftarrow$W/B. X_γ is changed from G to B if the number of its occurrences, called *score*, R_γ exceeds a *score threshold*, *S*, but $X_\gamma \leftarrow$ W if its *age* A_γ exceeds an *age threshold, M,* the age of γ is the time elapsed before its score is increased. *S* and *M* are two major design parameters that decide the detection sensitivity and false alarm rates of the system.

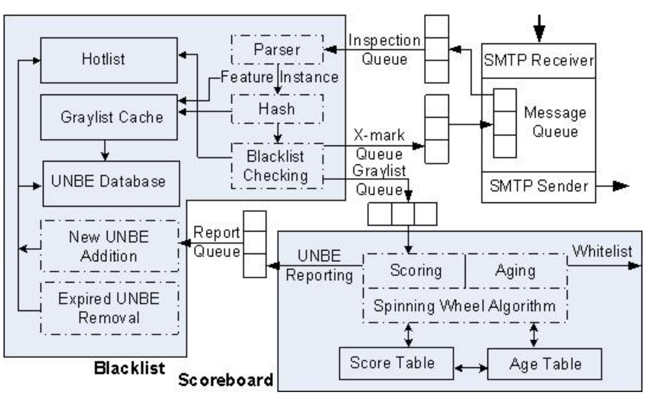

Figure 1 PEC Architecture.

Referring to Figure 1, we developed a cascaded filter architecture consisting of *blacklist* and *scoreboard* to track FIs. Messages being filtered are parsed for FIs by the feature parser in the blacklist module. After an FI γ is extracted and hashed to H_γ which is checked against the *hotlist,* the hash vector of currently active FIs of UNBE. If a hit occurs to H_γ on the hotlist, it means that γ is an UNBE instance, and the SMTP server could take countermeasure action, e.g., X-mark the (UNBE) message through the X-mark queue. Otherwise, (γ, H_γ) are placed into the *graylist cache,* and H_γ is placed into *graylist queue* of scoreboard for further tracking. The hotlist is essentially a single bit array with all or a part of H_γ as its address. The graylist cache serves as a temporary lookup table between FI and $H_{FI.}$ (γ, H_γ) is moved into the UNBE database once it appears on the *report queue*, i.e., R_γ was found to exceed *S* by the scoreboard. Or, (γ, H_γ) is simply removed from the cache when $A\gamma$ exceeds *M.* The average life span of UNBE feature instances is short. As such, a background thread periodically examines the UNBE database so that when it becomes cold for a certain period of time the FI can be removed and the computing resources recycled. H_γ passed from the blacklist to the scoreboard is tracked by *scoring* and *aging* functions, based on a *competitive aging-scoring scheme* (CASS), see Figure 2. If H_γ is new, it is placed into the *score table* (ST) and its score $R_\gamma \leftarrow 1$, and age $A_\gamma \leftarrow 0$ (in the *age table* (*AT*)). Otherwise, if H_γ is already in ST, R_γ is incremented and $A_\gamma \leftarrow 0$. In CASS, increasing of R_γ and reset of A_γ comes at the cost of aging of other entries, i.e., interlocked operations of increasing $R_{\gamma,}$ rest of A_γ, and increasing of $A_{\beta,}$ $\forall\ \beta \neq \gamma$, in one VC. An FI that does not have its score increased for a consecutive number of *VCs* is eliminated from the ST, i.e., $X_{FI} \leftarrow W$ and FI is not an UNBE feature. A critical design goal of CASS is to reduce the O(N) computing cost in age update of a naïve implementation to O(1). To solve this problem, we have developed a spinning wheel algorithm to keep track of ages of entries by modeling AT into a cyclic queue, and using the queue location of an entry to represent its age. Succinctly put, when an FI enters the scoreboard, the cyclic queue is spun by one position, so that the new head of the queue is that of the existing tail of queue, and the FI is placed at the new head location to overwrite the existing entry. A fixed number of steps are needed to maintain the interlocked relationship between AT and ST. PEC was implemented and tested on a Dell PowerEdge 1420 with Xeon 3.0 GHz CPU and 2GB memory. The scoreboard can process 1.2 M requests per second using randomly generated 32-bit unsigned integers. Depending on the pattern length, the throughput of the hotlist ranges from 300k to 900k/sec. We have developed an experimental environment, see Figure 3, that would allow the researchers to control the properties of UNBE traffic, based on the payload size, ratio of UNBE vs. regular emails, specific spamming corpus samples, etc. A control console was developed to display traffic flows and the detection dynamics. Multiple experiments were performed to test sensitivity of the detection algorithm.

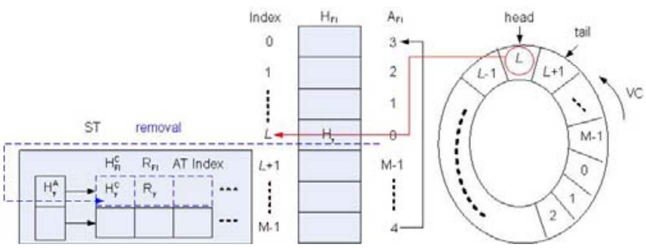

Figure 2. The spinning wheel algorithm in a CQ.

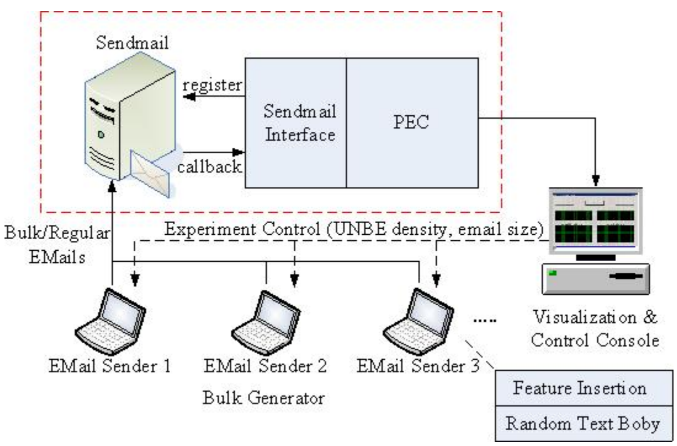

Figure 3. The server based experiment.

Packetized Filtering [7]

The original PEC architecture was tested on a sessionized server for its functionality. The second generation of PEC is called *packetized PEC*, or PPEC, for packet level filtering of messages. The PPEC consists of (1) an SMTP session manager, (2) a job scheduler, (3) feature parsers, and (4) a feature scoreboard. To keep track of the email messages in packet level, the PPEC uses the mechanism for SMTP session management. Once an email packet arrives at the zero-copying buffer, its sequence number is saved into a *packet list* to keep track of SMTP sessions. The sequence numbers are also used to restore the order of slightly out-of-order packets (typically 2). The feature scoreboard uses the competitive aging-scoring scheme (CASS) algorithm to detect surging of unknown patterns, while retiring others at very low computing costs. The architecture of the PPEC is illustrated in Figure 4.

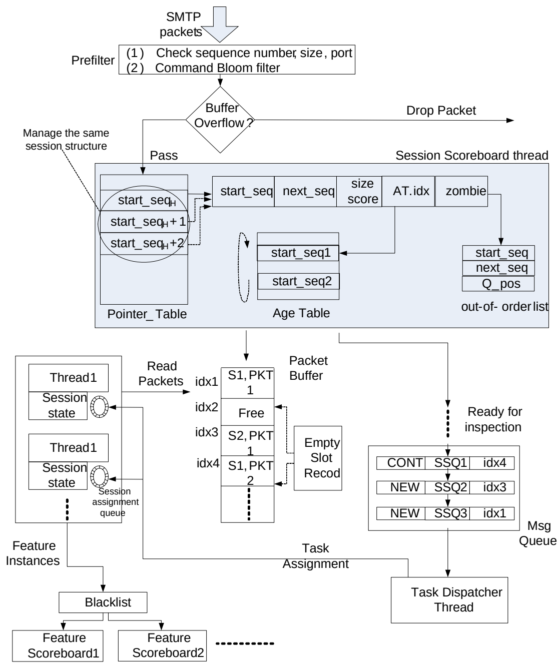

Figure 4. The Architecture of PPEC.

The testing environment of the PEEC consists of three subsystems, the PPEC, email generator, and the control console. All software modules in PPEC were implemented in *threads* and tested on a multi-core server. The thread based implementation is particularly important for parallel parsing of multiple features concurrently. We adopted the FLEX, a *deterministic finite automata* (DFA) based string parser, as the starting point for design of the *Packet based FLEX* (PFLEX). Extensively modified from FLEX, parallelized threads of PFLEX keep track of the state information of intermediate parsing results for packets acquired by the Pcap library. The email generator can generate emails based on user defined parameters, e.g., email densities, total volume, and ratio of regular and UNBE. To increase the benchmark volume, the generator can also interact with PPEC in one of the two simulated connection modes: (1) sending the SMTP packets directly through a POSIX socket, or (2) writing a series of SMTP sessions into a trace file. "URL" and "remote image source links" were extracted from TREC 2005 corpus and then combined randomly sampled web contents to generate the testing messages. The control console receives the detection report from the PPEC and displays it on a simple GUI.

Perl to POSIX Regexp format translation [2]

The SpamAssassin (SA) contains a large number of filtering rules based on known spamming techniques, but they must run in Perl. For line speed inspection, it is critical that we can transform those rules to the POSIX format so that they can be ported to different platforms. In addition to certain distributed inquiries for DNS, black lists, etc, a large portion of SA rules are loosely organized into different categories. By porting the Perl based SA rules to the POSIX format makes it easier to decouple the run-time scanners from the heavy-weight Perl run time libraries. a2px is a software tool to translate regular expressions in SA rules into the POSIX format. Sa2px has three-layer architecture:

1. The first layer translates plugins and special formats to their equivalent basic SA formats.
2. The second layer uses a syntax conversion approach to translate basic SA rules to the POSIX format. Syntax conversion bases on the predefined syntax translation mapping table.
3. The third layer is designed to group multiple rules, based on a backward grouping algorithm (BGA), so that they can be implemented into a DFA table using Flex or similar tools. BGA is an iterative greedy search heuristic which removes regexp having the largest interaction to other regexps in the group in one iteration. By using this method, BGA evaluates the interactive degree of rules to determine which rules to be grouped together. Then Flex standard input files (.lex) are automatically generated. After Flex compiling process, binary image source file (lex.yy.c) is constructed. Then the DFA transition table can be extracted from lex.yy.c, and binary scanner images are built as a side product.

Overall, sa2px can translate regexp in the whole rule set (uri, body, header, rawbody, and ReplaceTags plugin), and translation rate of 1115 SA regexp rules is 84.5%. In comparison, sa-compile can translate 296 of 453 body rules. The translated rules are then clustered into several main groups, except for some cases in which the regexp structures led to explosive state growth. Finally, DFA tables and (action number, rule name) pairs are generated. Experimental results show that the DFA table based implementation of these translated regexps cut down 66% of the execution time of the Perl (with sa-compile activated) based string scanning under process-level parallelization environment. Experiments on public spam corpus Trec2007 showed that the DFA generated by the sa2px-Flex tool chain correctly flags spamming contents, and the rule hit distribution of the SA regexp and their translated versions for different detection goals, i.e., adult, medication, education etc., are consistent.

Modular architecture for constrained character repetitions [3]

A major issue related to design of gateway content scanning engines is scalability and portability of different security management tools to a modular architecture. Given that the SA2PX tool can translate Perl based SA rules into the POSIX format, so that they can be compiled into DFA tables, we explore the design of a modular NFA-architecture for high performance regexp scanning. We have developed a modular architecture for implementation of Character class Constraint Repetition (CCR). The type of "At Least, Exactly, and Between" CCR patterns often lead to explosive growth of the DFA table size, and duplicated states in NFA.

The modular architecture (see Figure 5) includes a memory-based character class (the symbols that are acceptable by CCR) and a MIN-MAX counter pair (to count matching occurrences and check it against constraint repetition of CCR). We have developed an algorithm to resolve the ambiguity between two adjacent CCRs, e.g., [a-zA-Z]{3,6}[A-Z0-9]{2,4}. Moreover, an add-on checkpoint memory is proposed to enable the overlapped matching detection. It is notable that the character class, as well as all internal counters, could be configured through regular memory writes, and thus will be extremely easy to update.

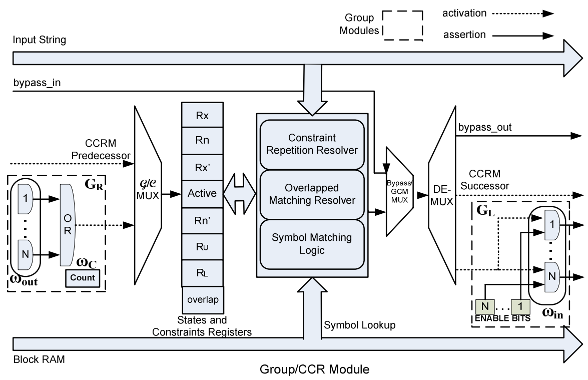

Figure 5. The architecture of a CCR Module

A modular architecture can drastically reduce the computing time overhead for compilation/synthesis of DFA/NFA systems. It takes hours to sort and combine NFA rules so that they can be efficiently compiled into DFA tables. For hardware based NFA implementations it routinely takes many hours to days to re-synthesis, map, place and route in order to change the layout of the NFA engine. We have developed a tool chain to automate the process of analyzing rule set, parsing it and generating syntax tree, mapping it to CCR interconnection network, and downloading the whole design to FPGA chip. We can design one type of topology that is optimized for regexps with long concatenation, and another one that is customized for regexps with lots of alternations. Experiments have been carried out on Virtex 5 LX110T device, and our results show that it can host up to 5000 CCRs (approximately 300 Snort regexp rules depending on the lengths of rules), and a throughput up to 3.616 Gbps. Parsing of rule sets to their implementation, configuration of the CCR interconnection network could be done within seconds.

Obfuscation Analysis [4]

Spammers use email spamming tools, such as Send-Safe, to obfuscate keywords in emails to evade spam filters. Obfuscation schemes based on random insertion and substitution make it very difficult to capture the morphed patterns. The roughly estimated 6×10^{19} ways to (mis-)spell "Viagra" was based on *substitution* and *insertion* on the six alphabets of the original word. We have developed a three phase statistics-based scheme, which is illustrated in Figure 6, to identify the spam words. The first phase is to group and sort the message possibly sent from the same campaign, and then identify the invariants and variants (possible obfuscation patterns). In the second phase, we propose two schemes, the location-counting (L-C) algorithm and transition chain, to recover the original spam words from the collected variants. The third phase is to approximately recover the obfuscation scheme. Major advantages of our scheme include that (1) it does not rely on existing lexicons, (2) it is language independent, and (3) no training is needed.

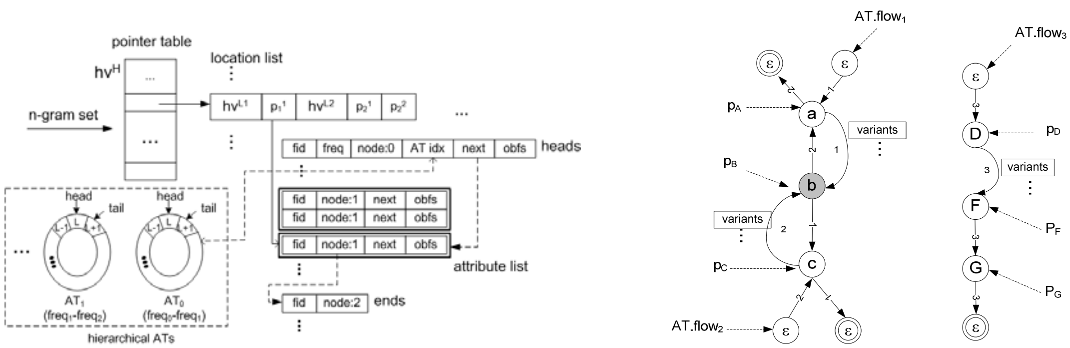

Figure 6. (a) data structure for campaign sorting, (b) graph representation of three campaigns

A sketch of the data structure for distinguishing the email campaigns and identifying the invariants and variants for the received messages is drawn in Figure (a). The three parts in the data structure are “pointer table” (for indexing of locations of n-grams extracted from the received message), “attribute list” (for storing the attributes of n-grams), and a hierarchical age tables (for tracking of the ages of campaigns). Figure (b) shows an abstract representation of three-campaign sorting, a→b→c, c→b→a, and D→F→G. The key procedures to operate the data structure are (1) new campaign initialization, (2) campaign merge, (3) retirement of inactive campaigns, and (4) collection of possible obfuscation patterns (variants).

After collecting a set of possible obfuscation patterns (variants) flanked by a pair of identified invariants, we propose two schemes to infer the spam words. The first is a *Location-Counting (L-C)* algorithm, based on a majority voting scheme to infer the most likely string from its obfuscated forms. The second scheme is based on the notion of a *Transition Chain*. In this scheme, we first construct a transition graph for all scanned symbols from the collected samples, and then choose the transition which has the highest probability based on an indentified symbol belonging to the spam words.

Summary and Future Work

In summary, the research exploration finds that gateway level ingress content inspection has the unique benefit of being able to collect the highest level of pattern statistics at the least communication overheads. By running efficient scanning algorithms and task scheduling techniques on modern hardware platforms, we can develop advanced content inspection systems by proper separation of rule composition system from the run-time modules. Our experiments on different hardware platforms (commodity multicore servers, FPGA, DPI engines) show that each of them has its own unique strengths, and should be designed to work with other types of technologies to achieve overall system defense goals.

While the primary focus of this project is aimed at filtering of email messages for spam, we believe the developed solutions can also be tailored for many content inspection applications. The following list represents some of the immediate applications that can be expanded from our current work.

(1) Light-weight virtual machine suitable for DPI environment. We have recently developed an X86 emulator, which can be tailored for this application.
(2) Scanning techniques to detect binary executables embedded in (intentionally) mislabeled files. Every file type uses a set of predefined markers to define the semantics of its contents. When executables are embedded into a file, we can use systematic marker scan, combined with executable analysis techniques to identify them.
(3) Coordinated content scanning. Single site content scanning only provides local information. Global scanning information sharing will provide much more accurate and reliable outcomes.
(4) Low level processing engine design, they include, but not limited to, approximate string matching [6], adaptive management of very large rule sets, multi-scale, cross-layer statistics analysis.

Take the (4) as an example, we can expand the scoreboard into a multistage architecture (see Figure 7) on Bivio or similar architectures for various applications, such as tracking of CP sessions for cross layer statistics analysis [5]. For instance, the Bivio Network Processor Unit (NPU) or other similar traffic sensors can capture the two way packets across a link, one can collect the different packet types of TCP sessions (SYN, ACK, FIN, RST, etc), so that one can measure and monitor the congestion control behaviors of a TCP session. This will help determining the portion of offending traffic sources through a gateway. The multistage scoreboard divides the time intervals between into quanta so that events in each stage are those of similar time periods earlier. This way, any events of relevance, for instance, recurring interactions between two nodes within an enterprise network and some outside node that can be extracted by the string scanners can also be captured by the multistage scoreboard.

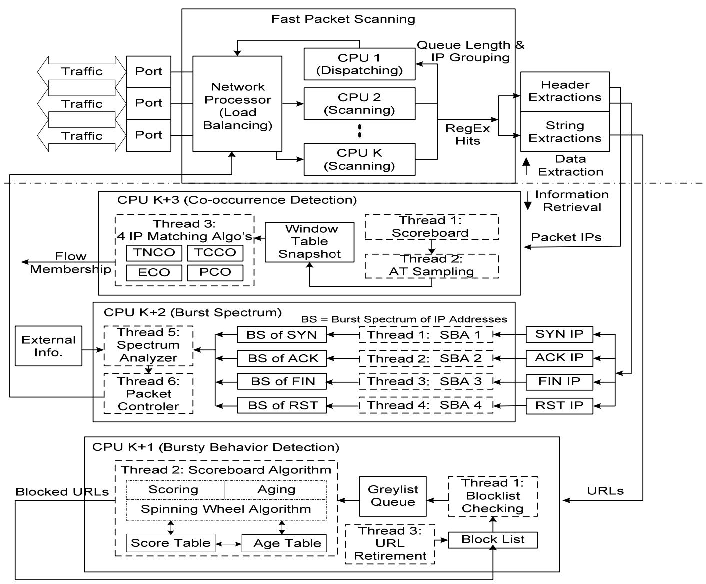

Figure 7. The multi-stage scoreboard architecture on Bivio.

References

1. Sheng-Ya Lin, Jonas Tan, Jyh-Charn Liu, Michael Oehler, "High-Speed Detection of Unsolicited Bulk Emails", the Symposium on Architectures for Networking and Communications Systems, Dec, 2007
2. Shi Pu, Cheng-Chung Tan and Jyh-Charn Liu, "SA2PX: A Tool to Translate SpamAssassin Regular Expression Rules to POSIX", 6th Conference on Email and Anti-Spam, 2009
3. Hao Wang, Shi Pu, Gabe Kneze, Jyh-Charn Liu, "A Modular NFA Architecture for Regular Expression Matching", accepted to ACM SIGDA FPGA conference, 2010
4. Cheng-Chung Tan, Jyh-Charn Liu, "Similarity-Based Sorting and Obfuscation Analysis of Message Streams", to be submitted
5. Shengya Lin, Jyh-Charn Liu, "On Classification of TCP Flows in the Middle of End-to-End Path", to be submitted
6. Hao Wang, Jyh-Charn Liu, "an NFA architecture for approximate string matching", under preparation
7. Cheng-Chung Tan, Jyh-Charn Liu, "Packetized string processing: architecture and performance", to be submitted